Towards location-aware mobile web browsers

Stephan Karpischek  
Chair of Information Management  
Department of Management, Technology and Economics  
ETH Zurich, Switzerland  
skarpischek@ethz.ch

Fabio Magagna  
Management of Information Systems  
Department of Management, Technology and Economics  
ETH Zurich, Switzerland  
fmagagna@ethz.ch

Florian Michahelles  
Chair of Information Management  
Department of Management, Technology and Economics  
ETH Zurich, Switzerland  
fmichahelles@ethz.ch

Juliana Sutanto  
Management of Information Systems  
Department of Management, Technology and Economics  
ETH Zurich, Switzerland  
jsutanto@ethz.ch

Elgar Fleisch  
Chair of Information Management  
Department of Management, Technology and Economics  
ETH Zurich, Switzerland  
efleisch@ethz.ch

ABSTRACT

Location Based Services (LBS) promise interesting business opportunities. Today, most LBS are either implemented in hardware devices, or downloaded and installed by mobile phone users as software applications. Both approaches lead to scattered markets and hinder standardization. This paper suggests an alternative approach, which is to enhance mobile web-browsers with location information and implement LBS at the server-side. We define the design space for the location enhanced mobile web and present an implementation of a location enhanced web browser for the currently predominant mobile phone operating system, i.e. Symbian S60.

Keywords

Browser, Location, Location Based Services, Mobile Web

1. INTRODUCTION

Location-awareness is one of the drivers of the ubiquitous computing paradigm. With sales of hardware and software for Location Based Services (LBS) of about 175 million EUR in Western Europe in 2008, the market for LBS is rapidly growing and may exceed 560 million EUR by 2011[0]. The major driver of LBS is the GPS technology in mobile phones, where 40% of the smartphones shipped in Q2 2008 have GPS built-in[0]. Besides GPS, LBS can also be achieved by mapping GSM cell IDs or WiFi networks to geographic locations (e.g. Google or Skyhook Wireless) [0].

While most LBS are offering navigation and routing services, and come in the form of highly specialized hardware devices (e.g. from companies like Garmin or TomTom), a wide range of mobile phone software for LBS is rapidly entering the market [0]. Popular examples include friends' tracking and tracing (e.g. Google Latitude\(^1\)), tourist and city guides (e.g. Wikitude\(^2\)), and location-based safety advice (e.g. iSafe\(^3\)). Although all these applications offer specific functions to mobile phone users, and need to be downloaded and installed on the mobile phone "one-by-one", all of them are actually implementing the exact same process of accessing the device's location, transferring it to a server application, and receiving and displaying the location-based information. The wide range of mobile platforms (e.g. different Symbian editions, Blackberry, iPhone, Windows Mobile, Google Android, etc.) and software development environments, as well as the fast life cycle of mobile phones and their operating systems make such distribution approach an expensive task.

An alternative approach that is inexpensive yet effective is to add "location capabilities" to mobile web-browsers. Most mobile phones have mobile web applications; thus eliminating the need to install additional software on the mobile phone. Imagine that mobile web browsers support a standardized way of transferring location information to a server. In this case, a small snippet of code on the server side suffices for processing the client's location information. Adding location capabilities to mobile web-browsers facilitates LBS implementation with much less resources than the usual way of developing a specific LBS software application for a wide range of heterogeneous mobile devices.

While there are already location enhanced browsers for the Desktop and for some mobile operating systems, to the best of our knowledge, there is still none for Symbian OS. This is a big gap as Symbian OS is by far the leading smartphone operating system. In 2008, 52.4% of all shipped smartphones were running Symbian OS, 16.6% were running Research in Motion’s proprietary operating system, 11.8% were running Microsoft Windows Mobile, and 10.7% were running Mac

---

1http://www.google.com/latitude
2http://www.mobilyz.com/wikitude.php
3http://www.freefamilywatch.com/
In this paper, we define the design space for the location-aware mobile web in Symbian OS. We focus on solutions which represent the location information in the WGS84 coordinate format.

In view of the popularity of the S60 interface platform of Symbian OS (87.8% of all Symbian phones use S60)\(^4\), we aim to enhance the Symbian S60 web browser with the mobile client’s location information. We focus on the 3rd Edition of S60, using Nokia N95 device for testing. In this paper, we will also present our browser implementation and an example of a server-side LBS application.

2. DESIGN SPACE

In the web there are several options to transfer location information from client to server. We organize the different options in a two-dimensional design space (see Figure 1). The "x-axis" shows the place in the communication protocol stack where the location information can be added, which is either in the URI of the web resource [0], or in the HTTP header [0], or in the JavaScript Document Object Model.\(^5\) The "y-axis" shows the application which adds the location information, which is either the web browser itself, or a plug-in.

There is currently no plug-in available which adds location information to the URI or the HTTP header. The i-mode browser adds location information to the URI [0], and several experimental implementations on Windows Mobile and Desktop [0], [0] add a location header to the client’s HTTP request. Instead of adding the location object at HTTP level, the W3C Geolocation API specification [0] proposes to add a location object to the JavaScript Document Object Model on the client side. A website can then include JavaScript code to access the client’s location. This method is implemented in several available location-enhanced web-browsers and plug-ins.

In the following, we will briefly explain the plug-ins and browsers’ approaches.

2.1 Plug-ins

The plug-in approach is emerging on Desktop computers as well as on mobile devices. Today, there are two dominant plug-ins that can be used to determine a client’s location: Firefox Geode [0]. With Geode, the user can also decide if

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Plug-ins</td>
<td>-</td>
<td>-</td>
<td>Geode</td>
<td>Google Gears</td>
</tr>
<tr>
<td>URI</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Header</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>HTTP</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>JavaScript</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Figure 1: Design space of the location-aware web.

In view of the popularity of the S60 interface platform of Symbian OS (87.8% of all Symbian phones use S60)\(^4\), we aim to enhance the Symbian S60 web browser with the mobile client’s location information. We focus on the 3rd Edition of S60, using Nokia N95 device for testing. In this paper, we will also present our browser implementation and an example of a server-side LBS application.

2.2 Browsers

On mobile devices, Google Gears is already included in the default Android web browser.

The browser on GPS-enabled Blackberry devices also adds a JavaScript object for location information[0]. Unfortunately, this solution is not compatible to either the W3C Geolocation API specification, or Geode, or Google Gears.

Fennec is the mobile version of Mozilla Firefox. The first beta release for Maemo devices (Nokia N800/N810) supports the JavaScript API to get the mobile device’s location[0]. There is also a pre-alpha version of Fennec for Windows Mobile but it is currently only optimized for a single device (HTC Touch Pro). A beta release for Windows Mobile is expected later this year. Porting Fennec to Symbian is however still in a very early stage.\(^5\)

3. CONCEPT

As can be seen in the previous section, there are several options to transmit the mobile client’s location to the web server. Although the JavaScript solution is the predominant solution, we decided to develop an HTTP solution for the Symbian S60 3rd Edition default browser. Specifically, we decided to add the mobile client’s location to the HTTP header and introduce a custom HTTP header for this purpose. We opt for this design concept because it is very intuitive for web developers and relatively easy to implement on the client side as compared to the JavaScript-based W3C Geolocation API specification.

Adding the location with URI parameters like what has been done in the i-mode browser could interfere with the existing web applications which use similar names for the URI parameters. Developing a plug-in for the S60 browser is not possible as the S60 browser has no interface for plug-ins. We thus decided to create an application which wraps around a browser API and adds a location header to each web request. The location information is updated with every new request.

4. IMPLEMENTATION

We used Symbian C++ to implement the proposed solution for Symbian S60 3rd Edition. With the Browser Control API [0], a browser is embedded in our application to enable users to request and view web content.

In addition, we add the current location of the mobile device to the HTTP request header with the Location Acquisition API [0]. A location server is running on the Symbian S60 device. The application can request the location from

\(^4\)http://en.wikipedia.org/wiki/S60_(software_platform)
\(^5\)http://www.w3c.org/DOM

Figure 2: Geode user dialog.
this background process. Hence, the whole positioning process is independent from the underlying location technology.

The retrieved location information is then added with a custom HTTP header “User-Location” to the HTTP request in the form of WGS84 coordinates for latitude and longitude, and the accuracy of the location in meters. The location is updated with every new request.

Figure 3 shows usage of the User-Location header. On the S60 device, the location-aware browser application adds the User-Location header to the HTTP request. On the server side, PHP example code for getting the location is shown. The client application and the complete source code can be downloaded from: http://www.mbusiness.ethz.ch/lamb.

5. EVALUATION

The available solutions for location enhanced mobile web (Google Gears, Firefox Geode and Blackberry browser) use JavaScript objects to transfer location information from the client to the server. In this section, we discuss the differences of the JavaScript-based solution and our solution to add a location header.

To evaluate our implementation and compare it with the other available solutions, we created an example website to display symbolic names (city and country) for the current location of the mobile client. Our example website is available on http://cocoa.ethz.ch/lamb_user_location.php.

Our example setup includes three building blocks: the mobile client, a web server and a web service. The web server and web service can either run on the same server or on different servers. In our example, the web service was provided by a third party (geonames.org).

Figure 5 shows the information flow in the JavaScript-based solution: The mobile client requests a web resource from the web server, and web servers respond with a web site containing HTML and JavaScript. The JavaScript code is then executed on the mobile client, accesses the client’s location, and sends it to a web service which returns location based information in JSON or XML format. In this solution, processing is needed at the client side.

Figure 6 shows the information flow in our proposed solution which is based on HTTP location header. In this solution, the mobile client only communicates with the web server and sends its location information with every HTTP request. The server can then process this location information and request location based information from a web service. The web server controls the communication with the web service and transfers only HTML to the client.

Our evaluation shows that the server side LBS implementation with location information in the HTTP header can achieve similar results as the solution based on JavaScript only for services which require mobile client’s location once per request. Figure 4 shows no visible difference for the user in the location-based information.

Our solution shifts the task of processing the location information from the mobile client to the server side. This has some advantages: The mobile web browser does not need to execute any code and is only used to render the HTML and display the content. This saves resources on the client side and most importantly, makes LBS available for mobile phones with less processing power or memory, and lightweight mobile web browsers.

The location header is relatively easy to add to mobile web browsers and browser plug-ins. Only a few lines of additional code are needed to implement our solution for the S60 web browser. So, the existing plug-ins could be easily extended to add the header in addition to the JavaScript object. Most web programming languages and frameworks (e.g. PHP, Ruby on Rails, Python) offer easy solutions for processing HTTP headers. Therefore the server side implementation is trivial for experienced web programmers.

On the other hand, a JavaScript-based approach allows for location updates on the client side without additional HTTP requests. This is a clear advantage when the application needs to react fast to location changes, e.g., in tracking...
or tracing applications, or when the current location is displayed on a map. However, in many LBS scenarios the user queries once for the current location and there is no need to update the location.

Privacy plays an important role in LBS: With the client sending location information to a web service, the decision if and how precisely to disclose the current location can be done during runtime for every service. When the web server queries one or many web services, this decision is done only once during implementation of the web site. In both cases the user can decide whether to send his/her location information or not. The individual and perceived level of control can be more fine-grained with a JavaScript-based solution.

We observed that there is no implementation of the W3C Geolocation API on Symbian devices so far. One reason could be that the JavaScript implementation on S60 browsers is rather complicated to extend or replace. Furthermore, as no standard has evolved for the location enhanced web, it is perhaps timely to seriously question if the Geolocation API specification is really suited for the mobile web as it relies on mobile browsers which offer JavaScript functionality. Also, the fact that the existing implementations are not compatible with each other might be an indicator that the specification is too complex for a mobile web solution.

To achieve standardization with a HTTP location header, there are only few decisions to be made. First, the community has to decide on a header name. Second, they have to decide which format to use for the location. The header name could be user-location or geolocation, and the format could be WGS84 coordinates with accuracy in meters.

6. CONCLUSION AND OUTLOOK

LBS increasingly become a key feature of the mobile web. We showed that it is relatively easy and scalable to implement LBS with web applications than with proprietary mobile phone software applications. Location-enhanced mobile web-browsers can easily be available for all mobile platforms as demonstrated by a simple solution of adding a HTTP location header to Symbian S60 web browser. Using location header, a standard for the location enhanced mobile web can be more easily achieved than using JavaScript.

7. REFERENCES